Coding Conventions

The purpose of this document is to provide a set of prescriptive rules that pertain to how code is to be written in the project. Our conventions are following Google Java Style Guide and slides from SOEN 6441 lecture. Specifically, we made highlight and comments on Google Java Style Guide. This paper is extracting from Google Java Style Guide for more concise version.

1. Code layout

1.1 Indentation

Code must be indented according to its nesting level.

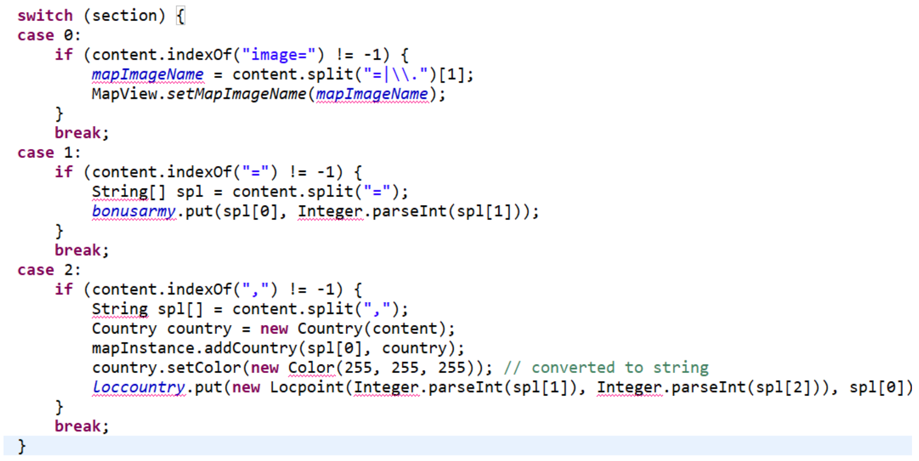
1.2 Formatting

1.2.1 Braces for K&R style

Braces follow the Kernighan and Ritchie style ("Egyptian brackets") for nonempty blocks and blocklike constructs:

- No line break before the opening brace.
- Line break after the opening brace.
- Line break before the closing brace.

Example for 1.1 and 1.2.1 :



```
switch (section) {
case 0:
    if (content.indexOf("image=") != -1) {
        mapImageName = content.split("=|\\.")[1];
        MapView.setMapImageName(mapImageName);
    }
    break;
case 1:
    if (content.indexOf("=") != -1) {
        String[] spl = content.split("=");
        bonusarmy.put(spl[0], Integer.parseInt(spl[1]));
    }
    break;
case 2:
    if (content.indexOf(",") != -1) {
        String spl[] = content.split(",");
        Country country = new Country(content);
        mapInstance.addCountry(spl[0], country);
        country.setColor(new Color(255, 255, 255)); // converted to string
        Loccountry.put(new Locpoint(Integer.parseInt(spl[1]), Integer.parseInt(spl[2])), spl[0])
    }
    break;
}
```

1.2.2 Vertical Whitespace (blank lines)

For readability purpose, blank lines can be added to separate code components/sections.

Where to put a blank line?

- between major sections of a long and complicated function.
- between public, protected, and private sections of a class declaration.
- between class declarations in a file.
- between function and method definitions.

Multiple consecutive blank lines are permitted, but never required (or encouraged).

Example:

```
public int getNumberPlayers() {
    return numberPlayers;
}

public void setNumberPlayers(int numberPlayers) {
    this.numberPlayers = numberPlayers;
}

public void addColor(Color color) {
    colors.add(color);
}
```

2. Naming conventions

- Package names are all lowercase, with consecutive words simply concatenated together (no underscores)
- Class names are written in UpperCamelCase.
- Method names are written in lowerCamelCase.
- Constant names use CONSTANT_CASE : all uppercase letters, with words separated by underscores.
- Nonconstant field names (static or otherwise) are written in lowerCamelCase.
- Parameter names are written in lowerCamelCase.

Example:

```
package controller;

public class StartupPhase

public void setupBorad() {
```

3. Commenting conventions

Commenting conventions that aim at increasing code understandability. However, comments should not provide information that can be easily inferred from the code.

3.1 Implementation comments style

Block comments are indented at the same level as the surrounding code. They may be in /* ... */ style or // ... style. For multiline /* ... */ comments, subsequent lines must start with * aligned with the * on the previous line. Example:

```
/*
while(!exit) {
    for (Player p : players) {
        reinforcementPhase = new ReinforcementPhase(p);
    }
}
*/

// if army available all used up, end phase
if(calculatedArmy==0) {
    this.finishMove = true;
}
```

3.2 Javadoc style

The basic formatting of Javadoc blocks is as seen in this example:

```
/**
 * This method is being used to check whether the Locpoint stack is not empty.
 * @param s The Locpoint stack.
 * @return The result, if not empty return true, if empty return false.
 */
```